Revista Internacional de Investigación e Innovación Tecnológica

Página principal: www.riiit.com.mx

A Heuristic proposal to minimize bottleneck and waiting time in a Flow Shop

Propuesta de heurística para minimizar el cuello de botella y el tiempo de espera en Flujo Continuo

Ireta-Sánchez, P.H., Martínez-López, R.*

Division of Posgraduate Studies and Research; Tecnológico Nacional de México/Instituto Tecnológico de Saltillo, Blvd. Venustiano Carranza No. 2400, Col. Tecnológico, Saltillo, 25280, Coahuila, México, C.P. 25280.
pedro.is@saltillo.tecnm.mx; ricardo.ml@saltillo.tecnm.mx*

Technological Innovation: Proposal of heuristics and relaxed model for batch processing solution.

Industrial application area: Batch processing lines**.**

Received: october 18th, 2022
Accepted: march 14th, 2023

Abstract

This article presents a Heuristic proposal to minimize two situations that occur when machines do not finish processing batches at the same time: 1) bottleneck when a machine takes more time to finish a batch and 2) waiting time that a batch must wait before being processed by the machine. To minimize both situations, batches must have the same processing time. To solve the above, the Heuristic proposal will select the job with the longest processing time and the number of pieces. Each piece will be a batch. The batch processing time is determined by the piece with the longest processing time. To validate the efficiency of the heuristic, information was collected from a case study of a metal-mechanic company. The Heuristic proposal was compared with other heuristics that have been used to solve the batch processing such as**:** First in-First Out (FIFO), Last in-First out (LIFO), Best Fit (BFF), First Fit (FF), Batch First Fit (BFF), First Fit Longest Processing Time (FFLPT), Decreasing Batch Size (DECR), Processing time to job size ratio increasing order (PIAI) and the Simplex method. The results showed an considerable reduction of bottleneck and waiting time using the Heuristic proposal compared with the others mentioned heuristic and the Simplex method.

Keywords: Optimization, Flow shop, Heuristic, Simplex, Metaheuristic.

Resumen

Este artículo presenta una propuesta de Heurística para minimizar dos situaciones que ocurren cuando las máquinas no terminan de procesar los lotes al mismo tiempo: 1) cuello de botella cuando una máquina toma más tiempo en terminar de procesar el lote y 2) tiempo de espera que un lote tiene que esperar para ser procesado por la máquina. Para minimizar ambas situaciones, los lotes deberán tener el mismo tiempo de procesamiento. Para resolver lo anterior, la propuesta de Heurística seleccionara el trabajo con el mayor tiempo de procesamiento y número de piezas. Cada pieza será un lote. El tiempo de procesamiento del lote es determinado por la pieza con el mayor tiempo de procesamiento. Para validar la eficiencia de la Heurística, se recolecto información de un caso de estudio de una compañía metal mecánica. La propuesta de Heurística se comparó con otras heurísticas que se han usado para resolver el procesamiento por lotes, tales como: Primera Entrada- Primera Salida (FIFO), Primera Salida-Primera Entrada (LIFO), Primer Ajuste (FF), Mejor Ajuste (BF), Primer Lote Ajustado (BFF), Primer Ajuste Tiempo de procesamiento más largo (FFLPT), Tamaño del Lote Decreciente (DECR), Orden creciente de la relación entre tiempo de procesamiento y tamaño del trabajo (PIAI) y Simplex. Los resultados mostraron reducción considerable del cuello de botella y tiempo de espera al usar la propuesta de Heurística comparado con las otras heurísticas mencionadas y método Simplex.

Palabras claves: Optimización, Flujo continuo, Heurística, Simplex, Metaheurísticas.

1. Introduction

In a batch scheduling problem (BPM), several jobs are grouped simultaneously on a machine where the processing time of the batch (makespan) is equal to the longest processing time of the job in the batch. According to *Ling and Wang* (2018) and *Chang and Wang (2004)* [1-2], BPM is used in manufacturing industries, for example, heat treatment, environment stress, food, pharmaceutical, etc. BPM has been applied as a policy because it has a significant economic impact on reducing costs, e.g., material handling, machine start-up, and line balancing (*Molla et al., 2014*) [3].

Since BPM was introduced as an NP-hard problem by (*Coffman et al., 1984*) [4], several methods have been proposed from an academic point of view (*Chandru and Uzsoy* (1993), *Bellanger et al., (2012), Fuchigami and S. Rangel* (2018) and *Mathirajan et al.*, 2014) [5-8]. Some methods like heuristics sort the jobs according to the processing time of the jobs and then generate the batches initialized; others sequence the jobs randomly, then the first job is placed in the batch that has enough space to accommodate it, and finally, another heuristic rule combines the manner to sort the jobs according to the processing time to fit the jobs to each batch then. *Ikura and Gimple* [9] introduced the BPM in the literature in 1986, they proposed the FirstOnly-Empty (FOE) algorithm to create batches ordering the jobs in non-decreasing way according to the release (r_j) time for identical job sizes (s_j) and a constant batch processing time. The authors mentioned that machine capacity,C, is the number of jobs that can proceed at the same time. The BPM becomes more complicated when the jobs have different processing times and**,** to solve it, the authors address two important decisions: 1) How to group jobs into batches? and 2) How to sequence the batches? (*Damodaran et al.,* (2012), and *Fowler and. Mönch* (2022)) [10-11].

At this point, it is necessary to establish that grouping jobs is an equivalent bin-packing problem, so the authors used the rule of first fit (FF) to solve the BPM, and the manner to generate batches depends on the environment of the production process. For a single machine, *Uzsoy* (1994) [12] cited distinct heuristics that use FF: Best Fit (BF), FF Longest Processing Time (FFLPT), FF decreasing job sizes (FFDECR), FF increasing the ratio of the processing time to the size of the job (FFPIA) and FF Shortest Processing Time (FFSPT); the objective function is to minimize the total competition time; the results indicated that the heuristics using LPT present good results. *Ghazvini and Dupont* (1998) [13] researchers proposed three heuristics based on the BF algorithm: Modified best Fit (MBF), BF Greddy Ratio (BFGR), and Dyna Algorithm on a single machine with different jobs size to minimize the mean flow time; the results show that Dyna presents results near to the optimal solution. *Li et al.*, (2005) [14] researchers suggested the Full-batch-longest-processing time rule (FBLPT) to solve a problem where ~~the~~ jobs can be split into different batches in a single batch processing machine with distinct jobs sizes; academics create the Schedule Split Algorithm where they create a short batch where they group the jobs with the same processing time; the paper does not present results. *Chen et al.*, (2011) [15] researchers present a Constrained Agglomerative Clustering of Batches (CACB) to minimize the makespan on a single machine and non-identical job size, where the first part, the jobs with identical processing time are clustered and the jobs with distinct processing time are grouped; the heuristic was compared with Genetic Algorithm (GA) and BFLPT; the outcome shows that CACB performs better than GA and BFLPT. *Li and Li* (2019) [16] present a performer heuristic based on BF and LPT where the jobs are identical in size and processing time, the machine has limited capacity, and the objective function is to minimize the makespan: Enumeration-based BFLPT decrease (EBFLPTD), the heuristic was compared with another heuristics: BFLPT, FFLPT and BFLPTD and the authors generated 100 random instances; the results indicated that EBFLPTD outperforms the other heuristics. *Miaomiao et al.*, (2020) [17] postulated 3 algorithms: A1, A2, and A3; to solve a batch processing machine with two limited capacity parallel machines, the jobs have the same processing time, a rejection penalty is applied, and the objective function is to minimize the makespan; the researchers conclude that their proposes are better than some existing bio-inspired algorithms.

In the case of the flow shop, *Johson* (1954) [18] propose a rule to obtain the optimal sequence in a flow shop by dividing the jobs in two sets; many researchers have proposed heuristics to solve the flow shop of batch processing machine (*Manjeshwar et al.*, 2009) [19]. *Tang and Liu (2009)* [20] present a heuristic, H, to generate batches to minimize the execution time with jobs that are identical; the manner to schedule them to the machine and transporter; the authors create their instances, and commented on the results show the heuristic performer to find an optimal solution. *Mirsanei et al.*, (2009) [21] established two heuristics: Acceptance/Rejection algorithm (ARA) and FLA algorithm which uses the BFF algorithm and LSPT rule with Simulated Annealing (SA) to minimize the makespan with non-identical jobs sizes.; they generated their instances; the results indicate that FLA algorithm with SA (FLSA) performs better than ARA with SA (ARSA). *Lin and Liao* (2012) [22] established three heuristics: Full Batch Earliest Due Date (FBEDD), Full Batch Family Shortest (FBFS), and Rolling FBFS (RFBFS) to solve a flow shop problem where the jobs have the same family and the batch processing time is common and the batch setup time is constant; the objective is to minimize the weighted sum of makespan, total completion time and total tardiness; to

prove the validity of the heuristic**,** a mixed integer programming (MIP) model was developed and random instances were created and the study case information was used; the results show that RFBFS is better than the other methods. *Baskar et al.*, (2018) [23] compared different heuristics based on the Nawaz, Enscore, Ham (NEH) algorithm to minimize makespan in a permutation flow show: NEHOA, NEHOAPSD, NEHO, NEHABXA; NEHABXB, NEHABXC, NEHABYA, NEHABYB, and NEHABYC; they used their instances. *Han and Lee* (2021) [24] researchers modified the NEH algorithm (MNEH) and contrasted it with GA and Iterated Greedy Algorithm (IGE), SA, and simplex method to minimize the makespan in a two-stage assembly flow shop with limited waiting time as a principal constraint; to test the proposal**,** random instances were generated, the results show that GA and IGE obtained good solutions for small and large problems than the other methods. *Lo and Lin* (2021) [25] researchers created two heuristics: JR-time permutation Heuristic and JR-resource Non-permutation Heuristic and compared them with Ant Colony Optimization to minimize the makespan in a flow shop with two different machines and non-identical jobs, they compared with random instances the heuristic and metaheuristics, the results indicate that JR- resource non-permutation obtained better makespan than the other methods. *Khalifa et al.,* (2021) [26] authors investigated multiple machines in a Flow Shop with the processing time, job weights, and break machines that are fuzzy. The researchers proposed a method that led to an optimal non-crossing sequence to minimize the total elapsed time under a fuzzy environment. The result indicates that the solution approach that no existing risk to applying the solution in a real-world problem because it is easy and simple to understand.

This brief review of the state of the art shows the different solutions methods and the way to order jobs to generate batches in a single machine and a continuous flow. Most of the methods are based on the First Fit heuristic to generate batches and send them to the processing line**.** They proposed some dispatching rules, for example, First in First Out, STP, LPT, and LIFO. Further, the authors assume that the sum of all the dimensions of the jobs in each batch cannot exceed the machine capacity, and always propose to minimize the same objective functions that mention by (*Fuchigami and Rangel*, 2018; *Fowler and Mönch, 2022*, and *Pinedo* (2016)) [7,11, 27]. This way of creating batches does not guarantee the machines finish processing them at the same time**,** which results in two situations that occur on the machines in the processing line 1) bottleneck: the machine needs more time to process a batch because it has a large processing time than the other one and 2) waiting time: the time that the batch is waiting before the machine process it. In both situations, a penalty is applied for each minute of bottleneck and waiting time generated.

Deriving from the above, it is important to consider other objective functions and restrictions that in the study case presented, for example, take it into account that some machines have enough space and consider the number of batches that a production line can process in a work shift.

Table 1 indicates the distinct heuristics and assumptions that different authors mentioned in their investigations to generate batches. The first column gives the name of the heuristic and the authors. The machine environment column indicates if the heuristic was used in a single machine (S) or a flow shop (F). The machine capacity column reveals if the heuristic considers (Y) or not (N) the capacity of the machine to generate the batches. The job size column considers if the sizes of all jobs are Identical or Non-Identical. Column Objective function specifies if the heuristic optimized one or more objective functions

mentioned by (Pinedo, (2016)). Column Batches same pt indicates if the heuristic created batches with the same processing time. Finally, the column Sort jobs shows the how the heuristic orders the jobs decreasingly (DEC)/Non-Dec or randomly (R).

Table 1. Assumptions that the methods used to create batches.

Heuristic	Machine environment		Machine capacity		Job size		Objective function		Batches same pt		Sort jobs	
	S	F	Y	N	Identical	Non-Identical	Y	N	Y	N	DEC/ NON-DEC	R
FF (Coffman et al. [4])	✓		✓			✓	✓					✓
BFF (Uzsoy 1998 [12])	✓		✓			✓	✓			✓		✓
FFLPT FFDECR FFPIAI FFSPT (Uzsoy 1998 [12])	✓		✓			✓	✓			✓	✓	
MBF, BFGR, Dyna (Ghazvini et al. [13])	✓		✓			✓	✓			✓		✓
FBLPT (Li et al., [14])	✓		✓			✓	✓			✓	✓	
CACB (Chen et al. [15])	✓		✓			✓	✓			✓	✓	
EBFLPTD (Li et al. [16])	✓		✓			✓	✓				✓	
A1, A2, A3 (Miaomiao et al. [17])	✓		✓			✓	✓			✓	✓	
H (Tang and Peng [20])		✓	✓		✓		✓			✓	✓	
ARASA, FLASA (Mirsanei et al. [21])		✓	✓		✓		✓			✓	✓	
FBEDD, FBF, RFBFS (Lin et al. [22])		✓	✓		✓		✓			✓	✓	
Distinct heuristic based on NEH (Baskar et al. [23])		✓	✓		✓		✓			✓	✓	

S=Single machine, F=Flow shop, Y=Yes, N=No, pt.=processing time, DEC=Decrease, NON-DEC= Non-decrease, R=Random.
*Source: Authors

Derive from above, this work presents a Heuristic proposal to generate batches that minimize the bottleneck and the waiting time for the continuous processing flow. This heuristic is based on the way jobs are painted in a metal-mechanical company in Saltillo, Coahuila. The company has a painting line with three cabins in continuous flow. Each job has the same processing time for all cabins. The cabins have enough space in square meters to process the jobs inside. The jobs are placed in an open area named work in process (WIP) to be sent to the first cabin using the First in First Out (FIFO) rule. The first cabin is the washing cabin where each job is sandblasted to

remove abrasive particles and send to the second cabin. The second cabin is where the jobs are painted in a single color, according to the production schedule, when the jobs are painted, they are sent to the drying cabin. The final cabin is where the jobs are drying. The problem begins when the programmer using the FIFO rule, he does not contemplate the following: 1) the FIFO rule sequences the jobs to the production line beginning with the first job that is at the top of the production schedule list without examinating that each job has a different processing time and number of units (elements); 2) each job can be split in a number of elements; 3) the jobs in the production schedule are painted with the same color; and 4) the cabins have enough space to combine different elements of other jobs. The above results in two important situations: 1) the bottleneck that is generated when a workstation takes more time to process a batch and 2) the waiting time that batches stay out at the workstation. Both situations generate an economic loss of about 5,000 USD per minute of bottleneck and 1,000 USD per minute of waiting time. Using the standard file notation established by (*Graham, 1979*) [28], the case study is denoted as: $F_m|p_j, s_j, batch|f$ where Fm is the Flow shop environment of the machine, p_j is the processing time of the job, s_j is the size of the job, batch all jobs grouped into batches and f is the objective function this case minimize the bottleneck and the waiting time.

To solve the two situations, the Heuristic proposed to generate batches that should have been finishing at the same time to be sent synchronously to the next workstation with the intention to minimize the bottleneck and waiting time. The Heuristic proposed to divide the job with the longest processing time into a number of pieces it had. Each piece is a batch, and the processing time of the batch will be the time it takes the piece to be processed. The rest of the pieces from the other jobs will be randomly selected to get into the batches, beginning with the first piece, then the second piece, and so on until the sum of their processing time does not exceed the batch processing time. Once all the pieces are in a batch, they are put together in "runs". The "runs" are the number of batches that will be processed in a work shift. The contribution of this work is not considering the machine's capacity to form batches as a principal constraint as the other methods use. It is necessary to mention that the use of work shifts, allows to know about the number of batches that will be processed during the workday. Also, the Heuristic proposed allows to identify which batches created the bottleneck and idle times to make solutions to prevent economic losses that affect the performance of the company. Finally, an adaptive linear programming model based on the BPM is given which considers the way that the Heuristic proposed generated the batches without considering the capacity of the machine and assuming that the machines have the same processing time, and the work shift is considered knowing the number of the batches that can be process in a work shift.

The remainder of the article is organized as follows: Section 2, a literature review; Section 3, a description of the mathematical model used; Section 4, a description of the methodology; Section 5, a computational experiment and Section 6, conclusions, and future work.

2. Materials and Methods

2.1 Batch processing machine

The manner that the methods mentioned in section 1, to generate batches, consider that once the process has started, it cannot be interrupted, and it is not possible to add or remove jobs to the batch until the process has been completed (*Kashan et al.,* (2006)) [29]. In addition, it is necessary to follow the assumptions mentioned by (*Zhou et al*., (2016), and *Li et al.,* (2013)) [30-31]: A set of n orders or jobs exists (j). Each job has a processing time (pj) and a dimension (sj).

Each machine has a processing capacity (C), and the sum of the dimensions of the jobs should not exceed the capacity of the machine. The jobs are grouped into batches b and form the set of batches to which they belong B. Thus, b ϵ B, the batch processing time will be the time of the job whose processing time is the longer $(P_b = max_{j\in b}\{p_j\})$. Finally, the objective is to minimize the total processing time (makespan).

Derived from the above, the model that has been used to solve the BPM as follows (*Li and Wang (2018), and Kashan et., al (2006)*) [1,29].

$$Min\ makespan = \sum_{b=1}^{n} P_{jb} \quad \textbf{(Eq. 1)}$$

Subject to:

$$\sum_{j=1}^{m} X_{jb} = 1, j = 1,2,\ldots,m \quad \textbf{(Eq. 2)}$$

$$\sum_{j=1}^{m} s_j X_{jb} \leq C \quad \textbf{(Eq. 3)}$$

$$P_b = max_{j\in b}\{P_j\} \quad \textbf{(Eq. 4)}$$

$$Cb_{max} \geq 0, P_b \geq 0 \quad \textbf{(Eq. 5)}$$

$$X_{jb} \in \{0,1\} \quad \textbf{(Eq. 6)}$$

Equation (1) represents the objective function. Equations (2), (3), and (4) state that all jobs must be part of a batch, the sum of the dimensions of the jobs in the batch must not exceed the capacity of the machine, and the batch processing time will be the processing time of the longest job in that batch. Equations (5) and (6) indicate the non-negative and binary values.

It is necessary to comment that the model presented below does not consider to produce batches with the same processing time. This results the presence of bottlenecks and waiting time in the processing line causes economic losses for the companies. In addition, the model does not consider the working shift time to know the number of batches that will be processed during the workday.

2.2 Adaptation model

To solve the problem described in section 1, an adaptation model was generated based on the BPM proposed by (*Li and Wang (2018)*), [1]. In the adaptation, variables, and restrictions are presented to generate batches that have the same processing time in order to minimize two situations that are responsible for the losses: 1) the bottleneck that is generated when a workstation takes more time to process a batch and 2) the waiting time that batches stay outside at the workstation. In addition, this model considers the following assumptions: once the process has started, it cannot be interrupted, pieces can be exchanged between batches before they enter the processing line or cabin, the batch processing time is determined by the piece that has the longest processing time in the batch; batches are grouped together to form production "runs"; the machine has enough space to process different pieces at the same time. Table 2 presents the symbols and notation used to solve the case study.

Table 2. Symbols and notations used in the relaxed model.

Symbols	Notation
J	Set of n jobs.
j_i	The job selected to form batches. The job with the longest processing time and number of pieces.
$j_i^{'}$	The jobs do not select to form batches.
J= $j_i + j_i^{'}$	All the jobs will be processed in the processing line.
PCS_i	A piece from j_i,.
$PCS_i^{'}$	A piece from $j_i^{'}$.
$TPCS_i$	The processing time for PCS_i.
$TPCS_i^{'}$	The processing time for $PCS_i^{'}$.
b	All PCS_i and $PCS_i^{'}$ are grouped in a batch.

TPb_i	The total processing time of the batch is equal to the processing $TPCS_i$.
P_r	Processing runs. All batches are group in runs.
W_s	Work shift.
TP_r	Processing time of the predecessor run.
TP_{r+1}	Processing time of the successor run.

*Source: Authors.

Derived from above, the model that has been used to solve the problem is as follows:

$$Min\ (TP_r - TP_{r+1}) \quad \textbf{(Eq. 1)}$$

Subject to:

$$J = j_i \cup j_i^{'} \quad \textbf{(Eq. 2)}$$

$$j_i = b_i \quad \textbf{(Eq. 3)}$$

$$\sum_{i=1}^{n} TPCS_i^{'} \leq Tb_i \quad \textbf{(Eq. 4)}$$

$$\sum_{b\in B} PCS_i = 1, \sum_{b\in B} PCS_i^{'} = 1 \quad \textbf{(Eq. 5)}$$

$$\sum_{b=1}^{n} b_i = P_r \quad \textbf{(Eq. 6)}$$

$$TP_r \leq W_s \quad \textbf{(Eq. 7)}$$

Equation (1) represents the objective function that it is the difference in processing time between the predecessor (TP_r) and the successor (TP_{r+1}), the production runs should be minimal. Constraint (2) is the set of all jobs that are in the production schedule. Constraint (3) explain the piece that has the largest processing time heads the batch. Constraint (4) indicates that the sum of the processing times of the pieces that do not lead each batch must not exceed the processing time that leads the batch. Constraint (5) ensures that each piece will be in a batch. Constraint (6) determines that all batches are grouped in a processing run. Finally, constraint (7) guarantees that the processing time of the run does not exceed the time of the work shift.

2.3. Heuristic proposed.

The Heuristic proposed selects the job with the longest processing time and a number of pieces. The selected job will be divided into pieces and each piece is a batch. The processing time of the batch is the processing time of the selected piece. The remaining jobs are divided into pieces and assigned randomly to the first available batch until the sum of their processing time exceeds the batch processing time. If no batches are available and there are still pieces that have not been assigned, a new batch is generated using the piece with the longest processing time. This process is finished when all the pieces are assigned in a batch. The batches are the added to create *"productions run"*. Each production run cannot exceed the time of the work shift.

Next, subtract the processing time of the predecessor production run and the successor run. If the result is a negative value, it is a bottleneck, otherwise, a waiting time is present. To minimize the bottleneck and waiting time, exchange elements between the runs that created the bottleneck. If the bottleneck decreases and the waiting time increases, continue with the exchange until the bottleneck is close to zero or zero.

The pseudo-code to generate batches is shown below:

Pseudo code of the proposed Heuristic.

START

(1) Select the job with the longest processing time J_i and number of pieces, PCS_i.
(2) Divide J_i in number of pieces PCS_i.
(3) Generate one batch b with one PCS_i.The batch processing time is the processing time of PCS_i.
(4) Divide the jobs, $J_i^{'}$, that are not selected in step 1 into elements $PCS_i^{'}$.

(5) Select the first available b and add pieces randomly from the elements $PCS_i^{'}$ while the sum of their processing time does not exceed the processing time of the batch.
(6) If the batch is full generate another batch and continue with step 5.
(7) If there are unassigned pieces $PCS_i^{'}$ in a batch, generate new batches using the $PCS_i^{'}$ with the longest processing time and continue with step 5.
(8) Sort the batches using the SPT rule.
(9) Group the batches in runs until their processing time does not exceed the work shift (W_s).
(10) Substract the processing time of the predecessor run (TP_r) and the successor run (TP_{r+1}). If the result is a negative value, it is a bottleneck. Otherwise, a waiting time is present.
(11) To minimize the bottleneck and waiting time, exchange elements between the runs $PCS_i^{'}$. If the bottleneck decreases and the waiting time increases continue the exhange until the bottleneck is zero or close to zero. If the waiting time decreases but the bottleneck increase stop the process.

END

Figure 1 shows the flow chart of the heuristic proposed to generate batches.

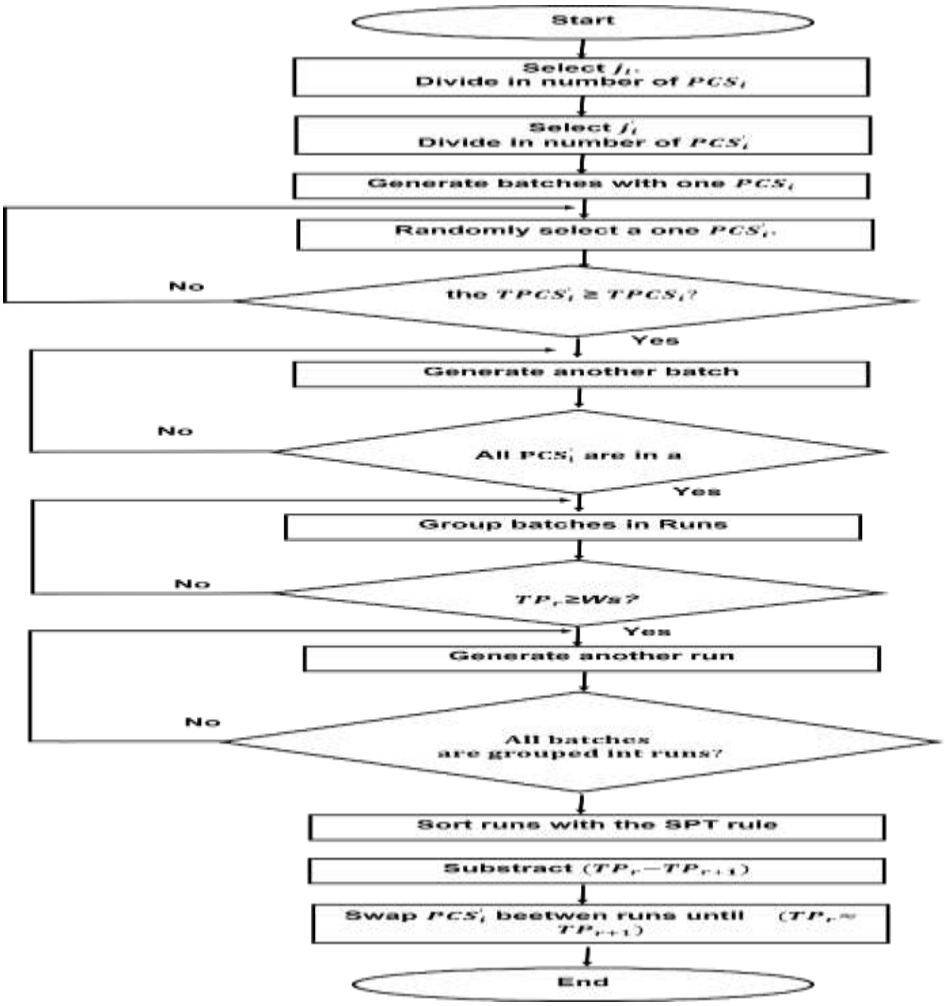

Figure 1. Flow chart of Heuristic proposed.

A comparison between the Heuristic proposed and different heuristics is presented below. The heuristics used in this example were FF, BFF, FLPT, and BFLPT and the programming linear model to generate the batches is mentioned in section 2.1. In this case, the machine capacity is 10 and the work shift is 20 minutes. Table 3 indicates the production schedule with the number of jobs that were used in this comparison. Column 1 is the number of jobs that will be processed in the processing line. Column 2 expresses the number of pieces for each job, in this case, the number of pieces for the job will be the size (s_j) that different authors used to generate batches. Columns 3 and 4 present the processing time for each piece and the total processing time for per job.

Table 3. Production schedule.

Job number	Number of pieces per job	Processing time per piece	Total processing time
J1	5	2	10
J2	3	3	9
J3	6	10	60
J4	4	5	20
J5	3	4	12

*Source: Authors.

The way in which the different heuristics are obtained is presented below:

- The First Fit (FF) heuristic consists of two steps: Step 1) Sequence the jobs randomly. Step 2) select the job at the head of the list and place it in the first batch with enough space to contain it. If the job does not fit in a batch, a new batch will be created. Repeat step 2 until all the jobs have been put in a batch.

- The Best First Fit is described as follows: Step 1) Sequence the jobs randomly. Step 2) select the job at the top of the list and move it to the batch with the lowest residual capacity (or the batch which is the fullest). In the case that the job does not fit in any existing batch, a new batch will be generated.

- The FFLPT heuristic can be described as follows: Step 1) Order the jobs in decreasing order by their processing time, and 2) Select the hob at the head of the list and send it in the first batch with sufficient space to contain it. If the job does not match in any batch, a new batch will be created. Repeat step 2 until all the job have been placed in a batch.

- The BFLPT heuristic is described as follows: Step 1) Organize the jobs in decreasing order by their processing time. Step 2) Take the job at the top of the list and deposit it in a reacheable batch that has the smallest residual capacity. If the job does not match any batch, a new batch will be created. Repeat step 2.

Table 4 displays the results of the four heuristics mentioned before. Rows 2 and 10 are step 1 which indicates in which manner each heuristic sequences the jobs. Rows 3 and 11 present step 2 and the result that the heuristics grouping the jobs to make the batches.

The results, obtained by the heuristic, show that all batches have different processing times. This happens because the heuristic takes as a main restriction that the size of the job in the batch cannot exceed the machine capacity. Also, as mentioned by

(*Damodaran and Chang (2007)*) [32] the aim of the BPM is to generate a minimum number of batches.

Table 4. Batches generate by FF, BF, FLPT and BFLPT.

FF			**BF**		
Step 1: random sequence: J2, J5, J1, J3, J4			Step 1: random sequence: J2, J5, J1, J3, J4		
Step 2			Step 2		
Batch	Jobs in the Batch	Processing time in minutes per batch	Batch	Jobs in the Batch	Processing time in minutes per batch
B1	J2, J5	12	B1	J2, J5	12
B2	J1	10	B2	J1, J4	20
B3	J3, J4	60	B3	J3	60
Total processing time $= \sum_{b=1}^{n} P_{jb}$		82	Total processing time $= \sum_{b=1}^{n} P_{jb}$		92
FLPT			BFLPT		
Step 1: decreasing jobs according of their processing time: J3, J4, J5, J1, J2			Step 1: decreasing jobs according of their processing time: J3, J4, J5, J1, J2		
Step 2			Step 2		
Batch	Jobs in the Batch	Processing in minutes per batch	Batch	Jobs in the Batch	Processing in minutes per batch
B1	J3, J4	60	B1	J3, J4	60
B2	J5, J1	12	B2	J5, J1	12
B3	J2	9	B3	J2	9
Total processing time $= \sum_{b=1}^{n} P_{jb}$		81	Total processing time $= \sum_{b=1}^{n} P_{jb}$		81

*Source: Authors.

In the case of the proposed Heuristic, it selects job 3 to establish the batches. Job 3 has six pieces with 10 minutes of processing time for each piece. The batch processing time is 10 minutes. To identify each piece, three-character were used: the first character indicates the number of pieces; the second character is the letter j which is an abbreviation of job, and the third character means the number of the job where the job comes. For example, 3J5 expresses: 3 elements from job 5; 1j7 is one element of job 7, and so on. Table 5 shows the number of batches created by the Heuristic proposed. Row 1 indicates the number of batches generated by the Heuristic proposed. Row 2 presents the piece with the longest processing time in the batch. Rows 3, 4, and 5 are the different pieces from the other jobs. Finally, row 6 is the batch processing time.

Table 5. Batches generated by the Heuristic proposed.

Number of batches	**B1**	**B2**	**B3**	**B4**	**B5**	**B6**
	1J3 PT: 10	1J3 PT: 10	1J3 PT: 10	1J3 PT: 10	1J3 PT: 10	1J3 PT: 10
	1J2 PT: 3	1J2 PT: 3	1J1 PT: 2	1J1 PT: 2	2J4 PT: 10	1J4 PT:5
	1J1 PT: 2	1J5 PT: 4	1J5 PT: 4	1J5 PT: 4		
	1J4 PT: 5	1J2 PT: 3	2J1 PT: 4			

Batch processing time $(P_b = max_{j \in b}\{p_j\})$	10	10	10	10	10	10

*Source: Authors.

As can be seen in the table above, the processing time of each batch is the same. This result was obtained because the Heuristic proposed does not consider the machine capacity as a restriction to generate the batches, such is the case of FF, BF, FLPT, and BFLPT. Instead, the Heuristic proposed makes batches with the same processing time finish their processing at the same time and send them synchronously to the next machine to avoid a bottleneck and waiting time. In addition, figure 2 shows the comparison of the total processing time. It can be seen that the Heuristic proposed has less total processing time than the others.

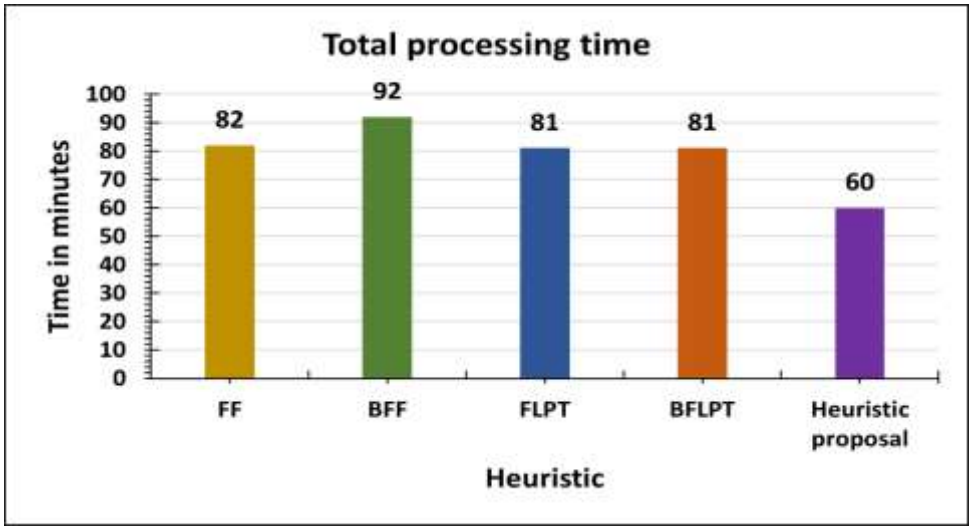

Figure 2. A comparative between the distinct heuristics and the Heuristic proposed.
Source: Authors.

3. Computational experiment

As mentioned in section 1, this problem is inspired by a real study case. Some companies combine distinct jobs because they have a common process, and the capacity of the machine (cabin) is not considered.

To verify the effectiveness of the Heuristic proposed, information was collected from the case study: 1) the dimensions in square meters of the WIP, wash and paint cabins; 2) a production schedule that shows the number of elements in each job, dimensions, and processing time of each one. Figure 3 shows the processing flow for the painting area. The WIP is an open area with enough space where the jobs are ordered using the FIFO rule. The jobs are painted in the same color. The next three cabins where the jobs are sandblasted, painted, and dried have enough space to process batches.

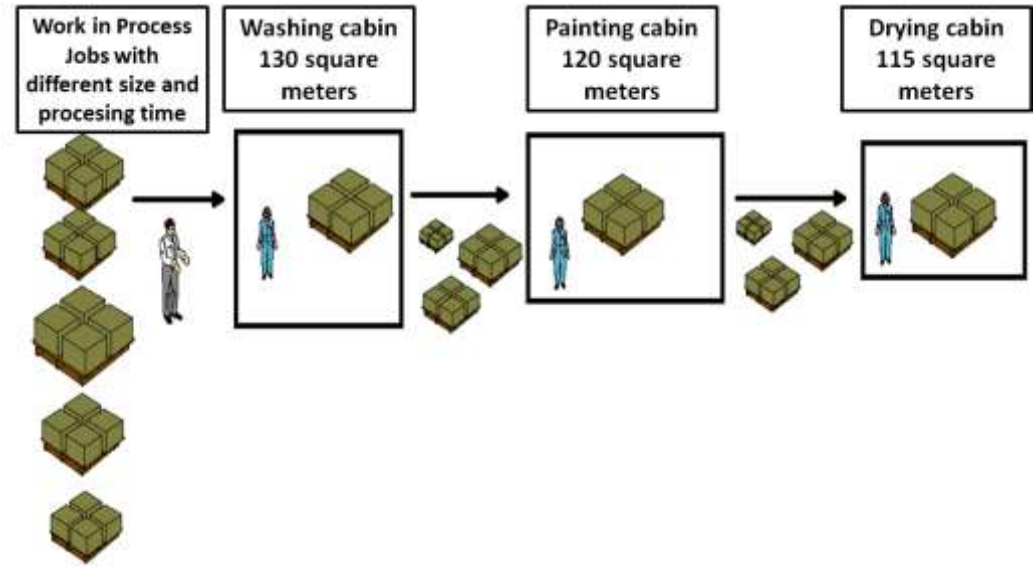

Figure 3. Processing flow. Source: Authors.

Table 6 shows the data collected in the framework of the study. Column 1 shows the order number, columns 2 and 3 are the dimensions and processing time of each piece of the job; the fourth column shows the number of piece of each job.

Table 6. Study case - Production Schedule.

Order Number	Dimensions Square Meters per piece	Process Time In minutes per piece	Quantity of pieces	Total Dimension	Total Time
j1	10.63	14.02	6	63.80	60.17
j2	9.83	13.23	10	98.33	132.30
j3-a	11.55	88.99	8	80.82	622.91
J3-b	11.55	88.99	7	92.37	711.89
j4	3.86	2.53	11	42.56	714.67
j5	7.11	9.25	15	106.67	1130.10

Source: Authors.

The capacity of the paint cabin is indistinct, and the work shift is 200 minutes. FIFO rule obtained -1069.35 minutes of bottleneck and 0 minutes of waiting time. This result occurred because the programmer introduces one job at a time. It is noteworthy to mention that with the FIFO rule the company loses $5,349,675 of dollars.

Table 7 shows the batches generated by the proposed heuristic before grouping them into runs. To understand the information, a vocabulary is presented: The column named Batch number indicates the number of the generated batch *B1, B2,B3;* the nomenclature, for example, J3 means one piece from job number 3, J4 indicates one piece from job 4 and so on. The column named PT per job means the processing time per job; the first number is the processing time of head of the batches and the rest of the numbers are the processing time of remainder pieces of the batch. For example, *B1* is the batch number one, J3 is one piece from job number 3 and the processng time for this job is 88.99 minutes (in this case it is the total processing time of the batch); J4 means one piece from job number 4 and the number 64.97 is the processing time for the piece.

Table 7. Generated batches with the proposed Heuristic.

Batch number	**PT per job**	**Batch number**	**PT per job**	**Batch number**	**PT per job**	**Batch number**	**PT per job**
B1		*B2*		*B3*		*B4*	
J3	88.99	J3	88.99	J3	88.99	J3	88.99
J4	64.97	J5	75.34	J4	64.97	J4	64.97
		J2	13.23				
Batch number	**PT per job**	**Batch number**	**PT per job**	**Batch number**	**PT per job**	**Batch number**	**PT per job**
B5		*B6*		*B7*		*B8*	
J3	88.99	J3	88.99	J3	88.99	J3	88.99
J4	64.97	J5	75.34	J5	75.34	J5	75.34
J1	10.03						
Batch number	**PT per job**	**Batch number**	**PT per job**	**Batch number**	**PT per job**	**Batch number**	**PT per job**
B9		*B10*		*B11*		*B12*	
J3	88.99	J3	88.99	J3	88.99	J3	88.99
J4	64.97	J5	75.34	J4	64.97	J1	10.03
Batch number	**PT per job**	**Batch number**	**PT per job**	**Batch number**	**PT per job**	**Batch number**	**PT per job**
B13		*B14*		*B15*		*J5*	*75.34*
J3	88.99	J3	88.99	J3	88.99		
J1	10.03	J5	75.34	J5	75.34		
J5	75.34						
Batch number	**PT per job**	**Batch number**	**PT per job**	**Batch number**	**PT per job**	**Batch number**	**PT per job**
B16		*B17*		*B 18*	*Time*	*B19*	
J5	75.34	J5	75.34	J5	75.34	J5	75.34
J4	64.97	J2	13.23	J4	64.97	J4	64.97
		J2	13.23				
Batch number	**PT per job**	**Batch number**	**PT per job**	**Batch number**	**PT per job**		
B20		*B21*		*B22*			
J5	75.34	J5	75.34	J5	75.34		
J2, J2, J2	39.69	J4	64.97	J2	13.23		
J1	10.3			J2	13.23		
				J2	13.23		
				J1	10.03		
				J2	13.23		
				J1	10.03		
				J1	10.03		

B=Batch, Time: processing time in minutes.
Source: Authors.

Using the Heuristic proposed as a solution method, 12 batches with 88.99 minutes of processing time and 7 batches with 75.30 minutes were obtained. The results allow us to know the number of batches that have different processing times and when they are grouped in run aids, which of them generate bottleneck and waiting time.

Figure 4 presents the processing time per run. Runs 1 through 7 have the same time because each run contains batches whose processing time is one of the pieces with the longest time. Run 8 has a processing time of 164.33 minutes because it was formed with two batches at different times, one with 88.99 minutes and the other with 75.34 minutes, respectively. Finally, runs 9 to 11 contain two batches with the same processing time of 75.34 minutes. The bottleneck obtained was -27.3 minutes and 0 minutes for waiting time. The results allow us to know the number of batches that have different processing times. When

grouped in runs, it helps them generate the bottlenecks and waiting time. With the Heuristic proposed, the company loses $27,300 dollars per minute generated by the bottleneck.

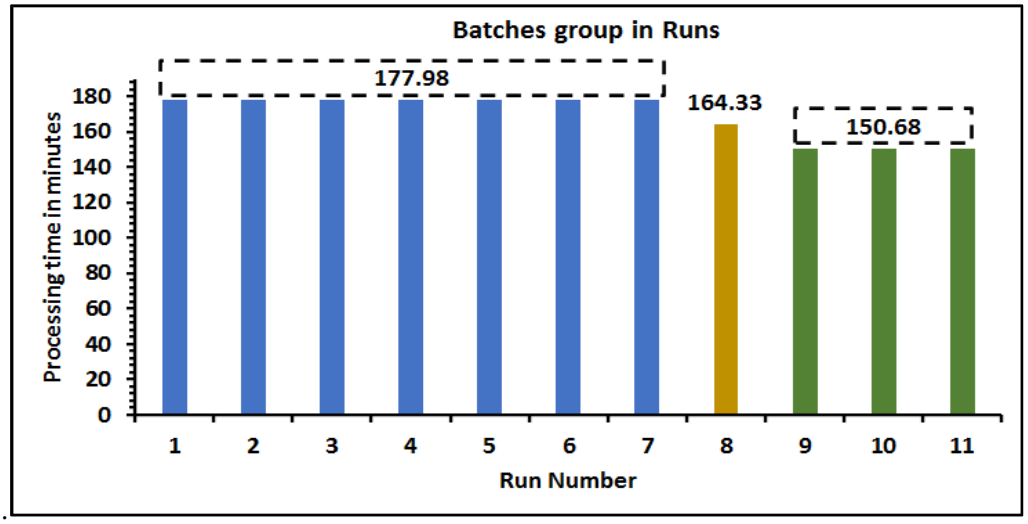

Figure 4. Generated runs.
Source: Authors.

Figure 5 illustrates a comparison of the processing times between the FIFO rule and the Heuristic proposed. It can be observed that when jobs are entered using the FIFO rule, the processing times are different because the rule does not group distinct jobs to form batches. The runs that have the same processing time is because the Heuristic proposed selects batches with the same time; otherwise, the runs that do not have the same time is because it has two or more batches with different time.

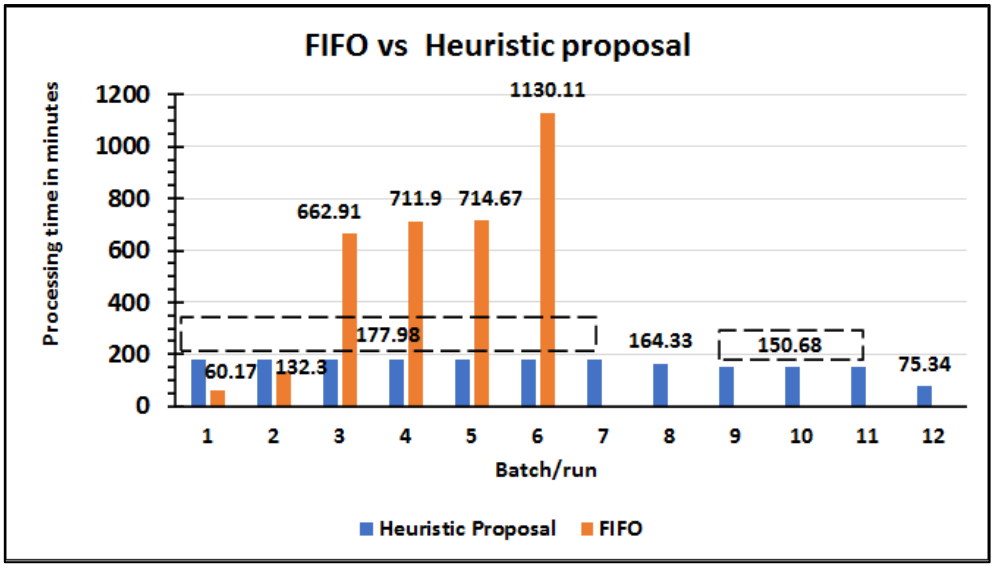

Figure 5. FIFO vs Heuristic proposed.
Source: Authors.

On the other hand, the Heuristic proposed was compared with the Simplex method using a commercial solver. The results obtained were nineteen batches: fourteen with 177.97 minutes, one batch with 153.96 minutes of processing time, three batches with 194.67 minutes, and another batch with 69.97 minutes. The results were obtained because Simplex performs combinations that comply with the

constraints of the problem: the batch processing machine and the duration of the work shift. Also, the results indicate that Simplex does not generate good results as mentioned by (*Lo and Lin (2021*) [25]. The Simplex method obtained -40.95 minutes of bottleneck and 153.95 minutes of waiting time.

In this case, Simplex method generated losses to the company in us$204,750, of bottleneck and us$153,950 of waiting time. Table 8 shows the results of the batches obtained by Simplex. The first column is the number of batches created; the second column is the number of pieces contained in the batch, and the third column is the processing time.

Table 8. Processing Time per Batch Obtained by Simplex.

Batch Number	Number of pieces	Processing time in minutes
1-14	3	177.97
15	1	153.96
16-18	1	194.97
19	1	69.97

Source: Authors.

As mentioned, the Heuristic proposed was compared with other heuristics that have been used by other authors to solve the BPM. Table 9 shows the result obtained by LIFO, SPT, LPT, BFF, FFLPT FFDECR, and FFPIAI using the data collected from the study case.

Table 9. Comparative between different heuristics.

Author	Method	Bottleneck	Waiting time	USD Bottleneck	USD Waiting time
[22]	LIFO	0	1069.93	0	$1,069,930
[22]	SPT	-1069.35	0	$5,346,750	0
[22]	LPT	0	1069.35	0	$1,069,930
[13]	BATCH FIRST FIT (BFF)	-1649.46	1741.23	$8,247,300	$1,741,230
[13]	FFLPT	-60.36	17.502	$301,800	$17,502
[13]	FFDECR	-42.86	0	$214,300	0
[13]	FFPIAI	-1274.63	620.13	$6,373,150	$620,13

Source: Authors.

The table shows that the result obtained by the SPT rule is the same as the result obtained by FIFO. This occurs because the SPT organizes the jobs from shortest to longest according to the processing times, and in this case, it matched the ordering by FIFO. The result of LPT was obtained because this method orders the jobs from longest to shortest, so the difference between the previous batch minus the next batch is always a positive number, which indicates that the order will be waiting for a certain time to be processed. For the rest of the rules, a random sequence is generated to create batches, and these result in different processing times, leading to the presence of bottleneck and waiting time. If these results are compared with those obtained by the Heuristic proposed, the latter reduced considerably the bottleneck and did not generate waiting time; both results are obtained because the Heuristic proposed from the beginning generates batches with each of the pieces that have the longest processing time and groups them in runs taking the duration of the work shift.

As a part of the validation of the Heuristic proposed, five random instances were generated. The value of each parameter was

as similar as possible to the collected data in the study case. It is necessary to mention that only one value was generated by each variable as mentioned by (*Talliard, 199*3) ([33]. The value of each parameter was considered a discrete variable with a uniform distribution. For this purpose, the parameters used are: Total processing time of each job $Uniform\ [1, \max T_t]$, machine dimension $Uniform\ [1, \max D_t]$, and number of pieces per job, $Uniform\ [1, \max C_\epsilon]$, where *max* is the maximum value of each of the parameters of the study case data.

Table 10 shows a comparison between the results of the waiting time and the bottleneck obtained by the Heuristic proposed and different dispatch rules (heuristic). The row is the name of the instance. The Method column indicates the name of the heuristic, and the Waiting time and Bottleneck columns presents the results in minutes by each heuristic. The number in the box indicates the total time obtained by the different solution methods. If the value is positive, it refers to a waiting time, otherwise it is a bottleneck.

Table 10. Results obtained Bottleneck and waiting time.

	Instance 1			**Instance 2**	
Method	Waiting time	Bottleneck	Method	Waiting Time	Bottleneck
H	92	0	H	0	-32
FIFO	126	0	FIFO	179	0
LIFO	0	-126	LIFO	0	-179
STP	126	0	STP	0	-37
LTP	0	-126	LTP	179	0
BFF	710	-126	BFF	179	0
FFLPT	126	0	FFLPT	179	0
	Instance 3			**Instance 4**	
Method	Waiting time	Bottleneck	Method	Waiting time	Bottleneck
H	20	0	H	3	0
FIFO	62	-44	FIFO	450	-270
LIFO	44	-62	LIFO	270	-450
STP	0	-42	STP	0	-335
LTP	160	0	LTP	335	0
BFF	62	-44	BFF	270	-129
FFLPT	42	0	FFLPT	270	0

	Instance 5	
Method	Waiting time	Bottleneck
H	41	0
FIFO	620	-1690.07
LIFO	1750.24	-620.13
STP	0	-1274.63
LTP	1274.63	0
BFF	787.04	-1202.51
FFLPT	1274.63	0

H: Heuristic proposed, BFF: Best First Fit, FFLPT: Longest Processing Time First Fit.
Source: Authors.

The results show that the Heuristic proposed does not generate a bottleneck for instances 1, 3, 4, and 5; in instance 2 LIFO, STP, LTP, BFF, and FFLPT have not obtained a bottleneck. For the waiting time only in instance 2, the Heuristic proposed did not present waiting time; for instance, 1 and 2 LIFO rule did not show waiting time; for instances 2,3,4, and 5, the STP rule did not generate any waiting time.

The results obtained for instances 1,3,4, and 5 indicate that the Heuristic proposed did not generate a bottleneck, which means that there is no penalty of USD 5,000 dollars per minute; otherwise, in instance 2, the proposed created a bottleneck, and it was penalized with USD 160,000 although it was less than the penalty of USD 895,000 and USD 185,000 by using LIFO and STP rules.

For the waiting time, the Heuristic proposed in instances 3, 4, and 5 generated lower values than most of the dispatching rules except with STP; these values indicate that the difference in processing time between the predecessor and successor runs is minimal. In the case of instance 1, although STP rule obtained a shorter waiting time compared to FIFO, STP, BFFA, and FFLPT, because it orders the jobs from the shortest to the longest processing time and add job by job to each batch until the batch is full. The results indicate that the time was not as close to zero, which indicates that the processing time of the predecessor batch is longer than that of the successor. In general, the Heuristic proposed allows for reducing the penalty of USD 1,000 per minute that a run must wait to be processed.

Lastly, the Heuristic proposed performed better than the other heuristics since it can form batches by dividing the job whose processing time is the longest of the whole production calendar, in addition, once the batches are formed, it makes runs considering the work shift time leading to know the total production time needed.

Conclusions and future work

In this article, a new Heuristic proposed to generate batches with the same processing time was proposed to help solve the BPM. In the same order, a new relaxed programming model was suggested. This model considers how to generate batches to minimize the bottleneck and waiting time.

The Heuristic proposed reduces the bottleneck in a 100% compared with FIFO rule. On the other hand, compared to the Simplex method, the Heuristic proposed reduces by 33.3% the bottleneck and 100% the waiting time. In the same direction, five non-deterministic random instances were generated based on the information collected from the case study. The results indicate that the Heuristic proposed compared to the LPT, SPT, LIFO, FIFO, BFF and FFLPT heuristics, minimizes the bottleneck and waiting time in an 82.7% and 72.4% respectively.

In conclusion, the Heuristic proposed generates batches with equal processing times if the selected job has enough pieces to head all batches without need to create new ones with other pieces. Also, it can be used in production process where the jobs can be separated in pieces. Finally, the heuristic allows to know the number of batches that can process in a work shift.

As a future work, it is proposed to include new restrictions to adjusts the model in a neutrosophic environment, as mentioned by (*Kumar and Khalifa (2021)*) [34], to establish a range of acceptance of economic losses causes by a machine failure, absence of operators, lack of material and transportation between different production area. Moreover, include some decision variables correlations, as mentioned in the investment portfolio by (*Kumar and Khalifa (2020)*) [35] , to give the advantages for the investors the decision to conclude a good return rate, risk loss rate and bank interest rate. Also, add human restriction,

for example**,** occupational safety and environmental care.

Finally, the use of other approximation algorithms such as metaheuristic will be studied.

Author Contributions

Conceptualization, P.I. methodology, P.I.; validation, P.I; writing-original draft preparation, P.I.; writing-review and editing, R.M.; supervision, R.M. The authors have read and agreed to the published version of the manuscript.

Acknowledge

The author wishes to thank to the Tecnológico Nacional de México/Instituto Tecnológico de Saltillo for the support obtained from the project 8479.20-P. Also, to the National Council of Science and Technology (CONACYT) for the scholarship granted.

Conflicts of Interest

The authors declare do not have conflict any conflict of interest.

References

[1] X. Li y Y. Wang, «Scheduling Batch Processing Machine Using MAX-MIN Ant System Algorithm Improved by Local Search Method,» Hindawi-Mathematical Problems in Engineering, vol. 2018, pp. 1-10, 2018, doi:10.1155/2018/3124182.

[2] P. Chang y H. Wang, «A heuristic for a batch processing machine scheduled to minimise total completion time with non-identical job sizes,» The International Journal of Advanced Manufacturing Technology, vol. 24, pp. 614-620, 2004, doi:10.1007/s00170-003-1740-9.

[3] S. Molla-Alizadeh-Zavardehi, R. Tavakkoli-Moghaddam y F. Hosseinzadeh Lotfi, «Hybrid Metaheuristics for Solving a Fuzzy Single Batch-Processing Machine Scheduling Problem,» The Scientific World Journal, vol. 14, pp. 1-10, 2014, doi: 10.1155/2014/214615.

[4] E. G. Coffman Jr., M. R. Garey y D. S. Johnson, «Approximation Algorithms for Bin-Packing — An Updated Survey,» Algorithm Design for Computer System Design, vol. 284, pp. 49-106, 1984, doi: 10.1007/978-3-7091-4338-4_3.

[5] V. Chandru y R. Uzsoy, «Minimizing total completion time on a batch processing machine with job families,» Operations Research Letters, vol. 13, pp. 61-65, 1993, doi:10.1016/0167-6377(93)90030-K.

[6] A. Bellanger, A. Janiak, M. Y. Kovalyov y A. Oulamara, «Scheduling an unbounded batching machine with job processing time compatibilities,» Discrete Applied Mathematics, vol. 12, pp. 15-23, 2012, doi:10.1016/j.dam.2011.09.004.

[7] H. Y. Fuchigami y S. Rangel, «A survey of case studies in production scheduling: Analysis and perspectives,» Journal of Computational Science, vol. 25, pp. 425-436, 2018, 10.1016/j.jocs.2017.06.004.

[8] M. Mathirajan, R. Gokhale y M. Ramasubrama, «Modeling of Scheduling Batch Processor in Discrete Parts Manufacturing,» the Supply Chain Strategies, Issues and Models, London., Springer-Verlag, 2014, pp. 153-192, doi:10.1007/978-1-4471-5352-8_7.

[9] Y. Ikura y M. Gimple, «Efficient scheduling algorithms for a single batch processing machine,» Operations Research Letters, vol. 5, nº 2, pp. 61-65, 1986, doi:10.1016/0167-6377(86)90104-5.

[10] P. Damodaran, D. A. Diyadawagamage, O. Ghrayeb y M. C. Vélez-Gallego, «A particle swarm optimization algorithm for minimizing makespan of nonidentical parallel batch processing machines,» International Journal of Advanced Manufacturing Technology, vol. 58, p. 1131–1140, 2012, doi:10.1007/s00170-011-3442-z.

[11] J. W. Fowler y L. Mönch, «A survey of scheduling with parallel batch (p-batch) processing,» European Journal of Operational Research, vol. 298, pp. 1-24, 22, doi: 10.1016/j.ejor.2021.06.012.

[12] R. Uzsoy, «Scheduling a single batch processing machine with non identical job sizes,» International Journal of Production Research, vol. 32, pp. 1615-1635, 1994, doi:10.1080/00207549408957026.

[13] F. J. Ghazvini y L. Dupont, «Minimizing mean flow times criteria on a single batch processing machine with non-identical jobs sizes,» International Journal of Production Economics, vol. 55, pp. 273-280, 1998, doi:10.1016/S0925-5273(98)00067-X.

[14] S. Li, G. Li, X. Wang y . Q. Liu, «Minimizing makespan on a single batching machine with release times and non-identical job sizes,» Operations Research Letters, vol. 33, pp. 157-164, 2005, doi:10.1016/j.orl.2004.04.009.

[15] H. Chen, D. Bing y G. Q. Huang, «Scheduling a batch processing machine with non-identical job sizes: a clustering perspective,» International Journal of Production Research, vol. 49, nº 19, pp. 5755-5778, 2011, doi:10.1080/00207543.2010.512620.

[16] X. Li, Y. Li y Y. Wang, «Minimizing makespan on a batch processing machine using heuristics improved by an enumeration scheme,» International Journal of Production Research, vol. 55, nº 1, pp. 176-186, 2016, doi:10.1080/00207543.2016.1200762.

[17] J. Miaomiao , . L. Xiaoxia y . L. Wenchang , «Single-Machine Parallel-Batch Scheduling with Nonidentical Job Sizes and Rejection,» Mathematics, nº 258, 2020, doi.10.3390/math8020258.

[18] M. Johson, «Optimal two- and three-stage production schedules with setup times included,» Naval Research Logistics Quarterly, nº 1, pp. 61-68, 1954, doi:org/10.1002/nav.3800010110.

[19] P. K. Manjeshwar, P. Damodaran y K. Srihari, «Minimizing makespan in a flow shop with two batch-processing machine using simulated annealing,» Robotics and Computer-Integrated Manufacturing, vol. 25, pp. 667-679, 2009, doi:10.1016/j.rcim.2008.05.003.

[20] L. Tang y P. Liu, «Two-machine flowshop scheduling problems involving a batching machine with transportation or deterioration consideration,» Applied Mathematical Modelling, vol. 33, pp.

1187-1199, 2009, doi: 10.1016/j.apm.2008.01.013.

[21] H. S. Mirsanei, B. Karimi y F. Jolai, «Flow shop scheduling with two batch processing machines and nonidentical job sizes,» International Journal of Advanced Manufacturing Technology, vol. 45, nº 553, p. 45, 2009, doi: 10.1007/s00170-009-1986-y.

[22] R. Lin y C.-J. Liao, «A case study of batch scheduling for an assembly shop,» International Journal of Production Economics, vol. 139, pp. 473-483, 2012, doi:10.1016/j.ijpe.2012.05.002.

[23] A. Baskar, M. A. Xavior y . N. Na, «Analysis of a Few Simple Heuristics for the Permutation Flow Shop Scheduling Problems for any Batch Processing Industry,» Materials Today: Proceedings, vol. 5, p. 11762–11770, 2018, doi: 10.1016/j.matpr.2018.02.145.

[24] J.-H. Han y . J.-Y. Lee, «Heuristics for a Two-Stage Assembly-Type Flow Shop with Limited Waiting Time Constraints,» Applied Sciences, vol. 11, nº 23, 2021, doi: 10.3390/app112311240.

[25] T.-C. Lo y B. Lin, «Relocation Scheduling in a Two-Machine Flow Shop with Resource Recycling Operations,» Mathematics, vol. 9, nº 1527, 2021, doi: 10.3390/math9131527.

[26] H. A. E.-W. Khalifa, S. S. Alodhaibi y P. Kumar, «Solving Constrained Flow-Shop Scheduling Problem through Multistage Fuzzy Binding Approach with Fuzzy Due Dates,» Advances in Fuzzy Systems , p. 8, 2021, doi: 10.1155/2021/6697060.

[27] M. L. Pinedo, Scheduling: Theory, Algorithms, and Systems. Fifth Edition, New York: Springer, 2016.

[28] R. L. Graham, E. L. Lawler, J. K. Lenstra y A. H. RinnooyKan, «Optimization and Approximation in Deterministic Sequencing and Scheduling: a SURVEY,» Annals of Discrete Mathematics, vol. 5, pp. 287-326, 1979, doi: 10.1016/S0167-5060(08)70356-X.

[29] A. H. Kashan, B. Karimi y F. Jolai, «Minimizing Makespan on a Single Batch Processing Machine with Non-identical Job Sizes: A Hybrid Genetic Approach,» de European Conference on Evolutionary Computation in Combinatorial Optimization, Budapest, 2006, doi: 10.1007/11730095_12.

[30] S.Zhou, M. Liu, H.Cheng y X. Li, «An effective discrete differential evolution algorithm for scheduling uniform parallel batch processing machines with non-identical capacities and arbitrary job sizes,» International Journal of Production Economics, vol. 179, pp. 1-11, 2016, doi:10.1016/j.ijpe.2016.05.014.

[31] X. Li, Y. Huang, Q. Tan y H. Chen, «Scheduling unrelated parallel batch processing machines with non-identical job sizes,» Computers & Operations Research, vol. 12, pp. 2983-299, 2013, doi.org/10.1016/j.cor.2016.08.015.

[32] P. Damodaran y P.-Y. Chang, «Heuristics to minimize makespan of parallel batch processing machines,» The International Journal of Advanced Manufacturing Technology, vol. 37, p. 1005–1013, 2007, doi: 10.1007/s00170-007-1042-8.

[33] E. Talliard, «Benchmarks for basic scheduling problems,» European Journal of Operational Research, vol. 64, pp. 278-285, 1993, doi: 10.1016/0377-2217(93)90182-M.

[34] H. E. Khalifa, P. Kumar y S. Mirjalili, «A KKM approach for inverse capacitated transportation problem in neutrosophic environment,» Sādhanā- Springer Nature, vol. 46, nº 166, 2021, doi: 10.1007/s12046-021-01682-5.

[35] P. Kumar y H. A. El- Wahed Khalifa, «Solving fully neutrosophic linear programming problem with application to stock portfolio selection,» Croatian Operational Research Review, nº 11, pp. 165-176, 2020, doi:10.17535/crorr.2020.0014.